

CÁC ĐỐI TƯỢNG DẠNG DANH SÁCH TRONG PYTHON

4.1. Iterator trong Python

4.1.1. Giới thiệu

- *Iterator* (hay *sequences types*) là các đối tượng gồm nhiều phần tử, *Iterator* cho phép ta truy cập đến từng phần tử và hành động này có thể được lặp đi lặp lại.

Về mặt kỹ thuật, *Iterator* trong *Python* phải thực hiện hai phương thức đặc biệt là `__iter__()` và `__next__()`, gọi chung là giao thức *iterator* (*Iterator Protocol*).

- Phương thức `__iter__` trả về chính đối tượng *iterator*. Phương thức này được yêu cầu cài đặt cho cả đối tượng "iterable" và *iterator* để có thể sử dụng các câu lệnh *for* và *in*.
- Phương thức `__next__` trả về phần tử tiếp theo. Nếu không còn phần tử nào nữa thì sẽ có lỗi ngoại lệ *StopIteration* xảy ra.

4.1.3. Các đối tượng dạng iterator trong python

Iterator	Có thứ tự	Sử dụng chỉ mục	Có thể thay đổi	Cho phép dữ liệu trùng
List	Có	Có	Có	Có
Tuple	Có	Có	Không	Có
Dictionary	Không	Có	Có	Không
Set	Không	Không	Có	Không

4.2. List

4.2.1. Giới thiệu

- Lists là tập hợp có thứ tự các phần tử (*elements*) thuộc nhiều kiểu dữ liệu khác nhau (như *strings*, *integers*, *objects*, *other lists*, ...). Do đó trong Python, key của một mảng có thể vừa là số, vừa là chuỗi (*associated array*). Tuy nhiên, nếu các phần tử đều thuộc cùng 1 kiểu dữ liệu sẽ giúp dễ

4.2. List

4.2.1. Giới thiệu

- *Lists* là tập hợp có thứ tự các phần tử (*elements*) thuộc nhiều kiểu dữ liệu khác nhau (như *strings*, *integers*, *objects*, *other lists*, ...). Do đó trong *Python*, *key* của một mảng có thể vừa là số, vừa là chuỗi (*associated array*). Tuy nhiên, nếu các phần tử đều thuộc cùng 1 kiểu dữ liệu sẽ giúp dễ xử lý hơn.
- *List* trong *Python* là cấu trúc mảng và các phần tử có *index* thứ tự tính từ 0 và *index* của phần tử cuối là số lượng phần tử trong list-1.

4.2.2. Khai báo

- Sử dụng cặp ngoặc vuông [] để khai báo một *list*.
- Khai báo *list*:
 - Khai báo *list* rỗng: `list1 = []`
 - Khai báo và gán giá trị
 - list2 = [9, 2.17, "Sai gon", "X"]
 - list3 = ["Ha noi", "Hue", "Sai gon"]
 - list4 = [7, 2, 5, 8, 1, 9]
 - Khai báo và gán giá trị dựa vào hàm *range(Start, Stop[Step])*
 - Ví dụ 4.5: `list5=range(2,11)`
 - Kết quả sẽ tạo ra 1 *list* với các giá trị như sau: [2, 3, 4, 5, 6, 7, 8, 9, 10]
 - Lưu ý:
 - Số lượng phần tử có trong *list* = *Stop* - *Start* (trong ví dụ đang xét là 11-2=9).
 - Giá trị lớn nhất trong *list* sẽ là *max-1*.
 - Khai báo và gán giá trị dựa vào dấu hoa thị (*)
 - Toán tử * đại diện cho "*phần còn lại*"
 - Ví dụ 4.6

- Giá trị lớn nhất trong *list* sẽ là *max-1*.

- Khai báo và gán giá trị dựa vào dấu hoa thị (*)

▫ Toán tử * đại diện cho "phần còn lại"

▫ Ví dụ 4.6

Mã lệnh	Kết quả
*a, b, c = [1, 2, 3, 4, 5] print ('a=', a, '; b=', b, '; c=', c)	a=[1, 2, 3]; b=4; c=5
a, *b, c = (1, 2, 3, 4, 5) print ('a=', a, '; b=', b, '; c=', c)	a=1; b=[2, 3, 4]; c=5
a, b, *c = {1, 2, 3, 4, 5} print ('a=', a, '; b=', b, '; c=', c)	a=1; b=2 ; c=[3, 4, 5]
a, b, c = {1, 2, 3, 4, 5} print ('a=', a, '; b=', b, '; c=', c)	ValueError: too many values to unpack (expected 3) '''Lỗi vì số lượng biến bên vé trái và số lượng giá trị bên vé phải không bằng nhau'''

▫ Lưu ý: chỉ được dùng tối đa 1 toán tử * trong vé trái

4.2.3. Truy xuất phần tử trong List

- Để biết được số lượng phần tử của 1 *list*, có thể sử dụng hàm *len (listName)*.

- Để biết được số lượng phần tử của 1 *list*, có thể sử dụng hàm *len (listName)*.
- Ví dụ 4.7

Mã lệnh	Kết quả
<pre>lst=['Sai Gon', 'Hue', 'Ha Noi'] print(len(lst))</pre>	3

- Truy xuất từng phần tử của *list* bằng chỉ mục (*index*), với $0 \leq index \leq len(listName)$.

4.2.4. Xuất nội dung list ra màn hình

Ví dụ 4.8: cho $L = [5, 10, 15, 20, 25, 30, 35, 40]$. Có 2 cách chính để in List:

- Cách 1: In toàn bộ *list* trong một lần. Cách này lại chia thành 3 cách chi tiết như sau:

Cách	Mã lệnh	Kết quả
1a	print(L)	[5, 10, 15, 20, 25, 30, 35, 40]
1b	print(*L)	5 10 15 20 25 30 35 40

4.2.4. Xuất nội dung list ra màn hình

Ví dụ 4.8: cho `L = [5, 10, 15, 20, 25, 30, 35, 40]`. Có 2 cách chính để in List:

- Cách 1: In toàn bộ list trong một lần. Cách này lại chia thành 3 cách chi tiết như sau:

Cách	Mã lệnh	Kết quả
1a	<code>print(L)</code>	[5, 10, 15, 20, 25, 30, 35, 40]
1b	<code>print(*L)</code>	5 10 15 20 25 30 35 40

- Cách 2: In lần lượt từng phần tử trong *list* bằng cách sử dụng lệnh lặp *for* hoặc *while*.

Cách	Mã lệnh	Kết quả
2a	#sử dụng for duyệt từng giá trị có trong list for item in L: print(item, end=', ')	5, 10, 15, 20, 25, 30, 35, 40,
2b	#sử dụng for và index để duyệt từng giá trị có trong list for index in range(len(L)): print(L[index], end=' ')	5 10 15 20 25 30 35 40
2c	#sử dụng while và index để duyệt từng giá trị có trong list index=0 while (index< len(L)) : print (L[index],end=', ') index = index+1	5, 10, 15, 20, 25, 30, 35, 40,

4.2.5.2. Chèn phần tử vào vị trí được chỉ định trong list

- Phương thức `list.Insert(index, element)` để chèn phần tử `element` vào vị trí `index` của `list`.

Ví dụ 4.10

Mã lệnh	Kết quả
<pre>numList = [5, 9, 6] print (numList)</pre>	[5, 9, 6]
<pre>numList.insert(1, 100) print (numList)</pre>	[5, 100, 9, 6]

4.2.5.3. Nối 1 iterable object vào list hiện tại

4.2.5.3.1. Sử dụng phép cộng (+) hoặc nhân(*)

- Ví dụ 4.11

Mã lệnh	Kết quả	Điễn giải
Lst1 = [5, 9, 7] Lst2 = [6, 2]		
Lst3 = Lst1 + Lst2 print ('Lst3 =', Lst3)	Lst3 = [5, 9, 7, 6, 2]	Nối Lst1 trước Lst2
Lst4 = Lst2 + Lst1 print ('Lst4 =', Lst4)	Lst4 = [6, 2, 5, 9, 7]	Nối Lst2 trước Lst1
Lst5 = Lst1 + (Lst2 * 2) print ('Lst5 =', Lst5)	Lst5 = [5, 9, 7, 6, 2, 6, 2]	Nối 2 lần Lst2 vào list kết quả

4.2.5.3.2. Sử dụng phương thức extend

- Phương thức `listName.extend(iterableObject)` để nối các thành phần của 1 iterable khác vào *list* hiện tại.

- Ví dụ 4.12

Mã lệnh

```
fruits = ['apple', 'banana', 'cherry']
cars = ['Ford', 'BMW', 'Volvo']
points = (1, 4, 5, 9)
```

```
fruits.extend(cars)
```

```
print(fruits)
```

```
cars.extend(points)
```

```
print(cars)
```

Kết quả

```
['apple', 'banana', 'cherry', 'Ford',
'BMW', 'Volvo']
```

```
['Ford', 'BMW', 'Volvo', 1, 4, 5, 9]
```

4.2.6. Cập nhật giá trị cho phần tử trong list

- Cú pháp: `listName[index] = giá_trị`
- Ví dụ 4.13

Mã lệnh	Kết quả	Điễn giải
<code>Lst = [5, 9, 6] print (Lst)</code>	[5, 9, 6]	
<code>Lst[2]=100 print(Lst)</code>	[5, 9, 100]	
<code>Lst[3]=50 print(Lst)</code>	IndexError: list assignment index out of range	Lst chỉ có index từ 0 đến 2, không có index=3

```
temp.py ✘ Bai61_Tr13.py ✘ Bai63_Tr13.py ✘
 1
 2     Ai ngồi, ai câu, ai sầu, ai thảm
 3     Ai thương, ai cảm, ai nhớ, ai trông
 4     Thuyền ai thấp thoáng ven sông
 5     Đưa câu mái vẩy chạnh lòng nước non'''
```

6

```
7 word=input("Nhập từ cần đếm: ")
8 print("Câu 63A (có phân biệt HOA và thường): trong đoạn thơ trên có %d từ %s" %(s.count(word), word))
9 print("Câu 63B (KHÔNG phân biệt HOA và thường): trong đoạn thơ trên có %d từ %s" %(s.upper().count(word.upper()), word))
```

10

```
11 int
12 float
13 str
14 list
15 dict
16 set
```

11 tránh đặt tên
giống keyword

```
17
18 lst
19 myList
20 List
```

4.2.7. Kiểm tra sự tồn tại của một phần tử trong list

4.2.7.1. Kiểm tra dựa trên IndexError Exception

- Python cho phép truy xuất một phần tử bất kỳ (dựa vào *index*) của mảng. Tuy nhiên, nếu truy xuất đến một phần tử không tồn tại thì ứng dụng sẽ báo lỗi. Do đó, trước khi truy xuất một phần tử, cần kiểm tra xem phần tử này đã tồn tại hay chưa bằng cách sử dụng cấu trúc *try ... except IndexError*
- Ví dụ 4.14

Mã lệnh	Kết quả
ConGiap = ["Ty/", "Suu", "Dan", "Meo", "Thin", "Ty.", "Ngo", "Mui", "Than", "Dau", "Tuat", "Hoi"] index = 12 try: x = ConGiap[index] # hay if ConGiap[index] > 0: print ('Phan tu thu', index, 'co gia tri la:', x) except IndexError: # xử lý khi trong List không có index đó print ('List khong co phan tu thu', index)	List khong co phan tu thu 12

4.2.7.3. Tìm vị trí của một giá trị trong list dựa trên phương thức index của list

Phương thức `list.index(element)` giúp tìm vị trí (`index`) của `element` trong một `list`. Nếu tìm thấy sẽ trả về `index` của phần tử đầu tiên tìm thấy. Nếu không tìm thấy sẽ phát sinh `ValueError Exception`.

Ví dụ 4.16

Mã lệnh	Kết quả
<pre>myList = [123, 'Sai Gon', 45.78, 'Sai Gon', 'Viet Nam'] try: print ("Index for Sai Gon:", aList.index('Sai Gon')) except ValueError: print ("Sai Gon is not in myList")</pre>	Index for Sai Gon: 1
<pre>try: print ("Index for 122 : ", myList.index(122)) except ValueError: print ("122 is not in myList")</pre>	122 is not in myList

except ValueError: print ("Sai Gon is not in myList")	Index for Sai Gon: 1
try: print ("Index for 122 : ", myList.index(122)) except ValueError: print ("122 is not in myList")	122 is not in myList

4.2.7.4. Đếm số lần xuất hiện của một giá trị trong list

- Phương thức `list.count(element)` giúp đếm số lần xuất hiện của `element` trong `list`.
- Thường được dùng để kiểm sự tồn tại.
- Ví dụ 4.17: xóa tất cả các số 1 đang có trong list

Mã lệnh	Kết quả
<code>myList = [1,2,1,3,1]</code> <code>print ("Before delete 1:", myList)</code>	Before delete 1: [1, 2, 1, 3, 1]
<code>while myList.count(1)>0:</code> <code> myList.remove(1)</code> <code>print ("After delete 1:", myList)</code>	After delete 1: [2, 3]

4.2.8. Copy list

- Phương thức `list.copy()` trả về bản sao của list hiện tại.
- Thường được dùng khi không muốn làm ảnh hưởng đến bản gốc của list.

Ví dụ 4.18

Mã lệnh

```
myList = [1, 2, 3]
copyList = myList.copy()
copyList.append(4)
print ('myList=', myList)
```

Kết quả

myList= [1, 2, 3]

```
print ('copyList=', copyList)
```

copyList= [1, 2, 3, 4]

4.2.7.1. Kiểm tra dựa trên IndexError Exception

- Python cho phép truy xuất một phần tử bất kỳ (dựa vào index) của mảng. Tuy nhiên, nếu truy xuất đến một phần tử không tồn tại thì ứng dụng sẽ báo lỗi. Do đó, trước khi truy xuất một phần tử, cần kiểm tra xem phần tử này đã tồn tại hay chưa bằng cách sử dụng cấu trúc try ... except. IndexError Exception
- Ví dụ 4.14

Mã lệnh	Kết quả
ConGiap=["Ty/", "Suu", "Dan", "Meo", "Thin", "Ty.", "Ngo", "Mui", "Than", "Dau", "Tuat", "Hoi"] index=12 try: x = ConGiap[index] #hay if ConGiap[index]>0: print ('Phan tu thu',index,'co gia tri la:',x) except IndexError: # xử lý khi trong List không có index đó print('List khong co phan tu thu',index)	List khong co phan tu thu 12

Buoi05 > Bai2_Tr26.py

Project

temp.py × Bai1_Tr26.py × Bai2_Tr26.py ×

1: Project

- Buoi05 D:\TTTH_KHTN\Python270\Buoi05
 - Bai1_Tr26.py
 - Bai2_Tr26.py**
 - Bai49_Tr11.py
 - Bai61_Tr13.py
 - Bai63_Tr13.py
 - temp.py

- External Libraries
- Scratches and Consoles

```

1 lst= list(map(int,input("Nhập nhiều số cách nhau bởi dấu phẩy").split(",")))
2 print("List vừa nhập: ",lst)
3
4 tong=0
5 for item in lst:
6     tong=tong+item
7 print ("Cách 1: tổng các số có trong list=",tong)
8
9 print ("Cách 2: tổng các số có trong list=",sum(lst))
10

```

Bai1_Tr26 × Bai2_Tr26 ×

Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
 Nhập nhiều số cách nhau bởi dấu phẩy? 5,2,7,3
 List vừa nhập: [5, 2, 7, 3]
 Cách 1: tổng các số có trong list= 17
 Cách 2: tổng các số có trong list= 17

In[3]:

item = {int} 3
 lst = {list: 4} [5, 2, 7, 3]
 tong = {int} 17
 Special Variables

```
lst= list(map(int,input("Nhập nhiều số cách nhau bởi dấu phẩy").split(",")))
print("List vừa nhập: ",lst)

tong=0
for item in lst:
    tong=tong+item
print ("Cách 1: tổng các số có trong list=",tong)

print ("Cách 2: tổng các số có trong list=",sum(lst))

nho=lst[0]
for item in lst:
    if nho>item:
        nho=item
print ("Cách 1: số nhỏ nhất có trong list=",nho)

print ("Cách 2: số nhỏ nhất có trong list=",min(lst))
```

có trong list=

01 item = { i Windows De

Buoi05 > Bai3_Tr26.py

Project

Project ▾

- Buoi05 D:\TTTH_KHTN\Python270
 - Bai1_Tr26.py
 - Bai2_Tr26.py
 - Bai3_Tr26.py
 - Bai49_Tr11.py
 - Bai61&62_Tr13.py
 - Bai63_Tr13.py
 - temp.py

External Libraries

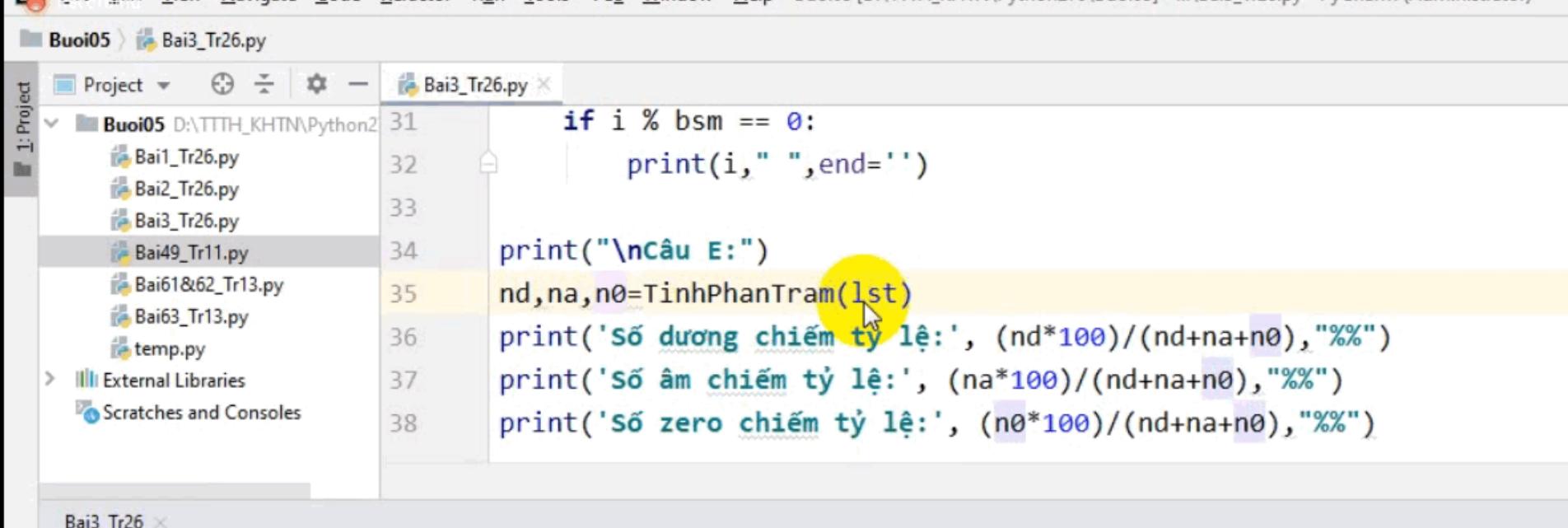
Scratches and Consoles

Bai3_Tr26.py

```
def TinhPhanTram(lst):
    nd = na = n0 = 0
    for i in lst:
        if i > 0:
            nd += 1
        elif i < 0:
            na += 1
        else:
            n0 += 1
    return nd,na,n0

lst = [1, -1, 2, 0, 5, 8, -13, 21, -34, 55, 87, 0]
print("List ban đầu: ",lst)
print("Câu B:")
lstC = []
print("Cac so <5:",end=' ')
for i in lst:
    if i < 5:
        print(i " " end='')
```

tính tỉ lệ phần trăm
số dương âm zero
trong list nhiều số



Bai3 Tr26

Câu D:

vui long nhap so m: >? 7

cac boi so cuu m: 0 21 0

Câu E:

Số dương chiếm tỷ lệ: 58.33333333333336 %

Số âm chiếm tỷ lệ: 25,0 %

Số zero chiếm tỷ lệ: 16.66666666666668 %

In[3]:

Buoi05 > Bai3_Tr26.py

Project

- Buoi05 D:\TTTH_KHTN\Python270
 - Bai1_Tr26.py
 - Bai2_Tr26.py
 - Bai3_Tr26.py**
 - Bai49_Tr11.py
 - Bai61&62_Tr13.py
 - Bai63_Tr13.py
 - temp.py
- External Libraries
- Scratches and Consoles

Bai3_Tr26.py

```

32     print(i, " ", end='')
33
34     print("\nCâu E:")
35     nd,na,n0=TinhPhanTram(lst)
36     print('Số dương chiếm tỷ lệ: {:.2f}%'.format((nd*100)/(nd+na+n0)))
37     print('Số âm chiếm tỷ lệ: {:.2f}%'.format((na*100)/(nd+na+n0)))
38     print('Số zero chiếm tỷ lệ: {:.2f}%'.format((n0*100)/(nd+na+n0)))

```

trong trường hợp nếu có nhiều thành phần ở format thì phải ghi số thứ tự ở đây. Nếu chỉ có 1 thành phần thì khỏi ghi



Bai3_Tr26

Câu D:

vui long nhap so m: >? 3

cac boi so cua m: 0 21 87 0

Câu E:

Số dương chiếm tỷ lệ: 58.33%

Số âm chiếm tỷ lệ: 25.0 %

Số zero chiếm tỷ lệ: 16.666666666666668 %

In[3]:

```

01 bsm =
01 i = {in
>   lst = {
>     lstC =
01 n0 = {
01 na = {
01 nd = {
>     Specia

```

5	XỬ LÝ NGOẠI LỆ (Exception Handling).....	133
5.1.	Lỗi cú pháp (Syntax Error).....	133
5.2.	Lỗi ngoại lệ (Exception Error).....	133
5.3.	Assertions	133
5.4.	Standard Exceptions	134
5.4.1.	Các exception có sẵn trong Python.....	134
5.4.2.	Sử dụng try ... except trong việc xử lý ngoại lệ.....	135
5.5.	Exception do người dùng định nghĩa (User Defined Exception).....	137

XỬ LÝ NGOẠI LỆ (Exception Handling)

5.1. Lỗi cú pháp (Syntax Error)

- Là các lỗi lập trình khi chương trình bị viết sai cú pháp quy định. Lỗi này còn biết đến như lỗi phân tích (*parsing error*).
- *Ví dụ 5.1:*

Mã lệnh	Kết quả
<pre>i=1 while True print(i) i=i+1 if i>=5: break</pre>	<pre>File "D:/MyProject/venv/exercise.py", line 2 while True ^ SyntaxError: invalid syntax</pre>

- Bộ phân tích lặp lại dòng gây lỗi và hiển thị một mũi tên nhỏ (^) trở vào điểm đầu tiên lỗi được phát hiện (lỗi nằm ở *phía sau* dấu hiệu mũi tên).
- Trong ví dụ trên, lỗi được phát hiện tại dòng lệnh *while*, ngay sau từ khóa *True*, vì thiếu một dấu hai chấm (:). Python hiển thị *Tên tập tin* và *số dòng* để hỗ trợ trong việc tìm lỗi.

Buoi05 > temp.py

Project

Buoi05 D:\TTTH_KHTN\Python2
Bai1_Tr26.py
Bai2_Tr26.py
Bai3_Tr26.py
Bai49_Tr11.py
Bai61&62_Tr13.py
Bai63_Tr13.py
temp.py

External Libraries
Scratches and Consoles

Bai3_Tr26 x temp x

```
a=5
b=0
print(a/b)
```

Bai3_Tr26 x temp x

File "C:\Program Files\JetBrains\PyCharm Community Edition 2019.3.4\plugins\python-
pydev_imports.execfile(filename, global_vars, local_vars) # execute the script
File "C:\Program Files\JetBrains\PyCharm Community Edition 2019.3.4\plugins\python-
exec(compile(contents+"\n", file, 'exec'), glob, loc)
File "D:/TTTH_KHTN/Python270/Buoi05/temp.py", line 3, in <module>
 print(a/b)
ZeroDivisionError: division by zero

In[3]:

Buoi05 > temp.py

1: Project

Buoi05 D:\TTTH_KHTN\Python270
Bai1_Tr26.py
Bai2_Tr26.py
Bai3_Tr26.py
Bai49_Tr11.py
Bai61&62_Tr13.py
Bai63_Tr13.py
temp.py

External Libraries

Scratches and Consoles

Bai3_Tr26 x temp x

File "C:\Program Files\JetBrains\PyCharm Community Edition 2019.3.4\plugins\python-
pydev_imports.execfile(filename, global_vars, local_vars) # execute the script
File "C:\Program Files\JetBrains\PyCharm Community Edition 2019.3.4\plugins\python-
exec(compile(contents+\n", file, 'exec'), glob, loc)
File "D:/TTTH_KHTN/Python270/Buoi05/temp.py", line 4

^

SyntaxError: unexpected EOF while parsing

In[3]:

5.3. Assertions *phản này báo qua*

- Cú pháp: **assert(Expression), [, Arguments]** thông_báo_lỗi
- Giải thích:
 - Khi thấy một *assertion statement*, Python đánh giá các *expression* kèm theo. Nếu *expression* có kết quả là:
 - *True*: lệnh đi ngay sau lệnh *assert* sẽ tiếp tục được thực hiện.
 - *False*: Python sẽ phát ra một *AssertionError Exception* bằng cách dùng *ArgumentExpression* làm đối số cho *AssertsError*.
 - Các ngoại lệ của *AssertsError* có thể được bắt và xử lý như bất kỳ ngoại lệ nào khác bằng cách sử dụng câu lệnh *try-except*, nhưng nếu không được xử lý, chúng sẽ chấm dứt chương trình và tạo ra một *traceback* (*module traceback* là cách khác để xử lý *exception* trong *Python*).
 - Về cơ bản, *traceback* được sử dụng để xuất dấu vết của một chương trình sau khi một *exception* xảy ra. *traceback* bao gồm thông báo lỗi, số dòng gây ra lỗi và *call stack* của *function* gây ra lỗi.

5.4. Standard Exceptions

5.4.1. Các exception có sẵn trong Python

TT	Exception Name	Mô tả
1	Exception	Đây là lớp cơ sở (base class) cho tất cả các exception, exception này sẽ xuất hiện khi có bất cứ một lỗi nào xảy ra.
2	StopIteration	Xuất hiện khi phương thức <i>next()</i> của <i>iterator</i> không trả đến một đối tượng nào.
3	SystemExit	Xuất hiện khi dùng phương thức <i>sys.exit()</i>
4	StandardError	Lớp cơ sở cho tất cả các exception, ngoại trừ <i>StopIteration</i> và <i>SystemExit</i> .
5	ArithmetError	Lớp cơ sở cho tất cả các lỗi xảy ra khi tính toán các số.
6	OverflowError	Xuất hiện khi thực hiện tính toán và giá trị tính toán vượt quá ngưỡng giới hạn cho phép của kiểu dữ liệu.
7	FloatingPointError	Xuất hiện khi tính toán các số kiểu float thất bại.

		dori tượng nào.
3	SystemExit	Xuất hiện khi dùng phương thức sys.exit()
4	StandardError	Lớp cơ sở cho tất cả các <i>exception</i> , ngoại trừ <i>StopIteration</i> và <i>SystemExit</i> .
5	ArithmetricError	Lớp cơ sở cho tất cả các lỗi xảy ra khi tính toán các số.
6	OverflowError ???	Xuất hiện khi thực hiện tính toán và giá trị tính toán vượt quá ngưỡng giới hạn cho phép của kiểu dữ liệu.
7	FloatingPointError	Xuất hiện khi tính toán các số kiểu float thất bại.
8	ZeroDivisionError	Xuất hiện khi thực hiện phép chia (<i>division</i>) hoặc chia lấy dư (<i>modulo</i>) một số cho 0 (<i>zero</i>).
9	AssertionError	Xuất hiện trong trường hợp lệnh assert thất bại.
10	AttributeError	Xuất hiện khi không tồn tại thuộc tính này, hoặc thiếu tham số truyền vào cho thuộc tính.
11	EOFError	Xuất hiện khi không có dữ liệu từ hàm input() hoặc raw_input() hay lỗi do theo tệp trên file khi con trỏ đang ở cuối file.

24	SystemError	Xuất hiện khi trình thông dịch phát hiện có vấn đề, nhưng lúc này trình thông dịch Python không tự thoát (kết thúc) được
25	SystemExit	Xuất hiện khi trong code không sử dụng hàm sys.exit() nhưng trình thông dịch Python vẫn được thoát bằng hàm sys.exit().
26	TypeError	Xuất hiện khi thực thi toán tử hoặc hàm mà kiểu dữ liệu bị sai so với kiểu dữ liệu đã định nghĩa ban đầu.
27	ValueError	Xuất hiện khi chúng ta build 1 function mà kiểu dữ liệu đúng nhưng khi chúng ta thiết lập ở tham số là khác so với khi truyền vào.
28	RuntimeError	Xuất hiện khi lỗi được sinh ra không thuộc một danh mục nào.
29	NotImplementedError	Xuất hiện khi một phương thức trừu tượng cần được thực hiện trong lớp kế thừa chứ không phải là lớp thực thi
30	UnboundLocalError	Xuất hiện khi chúng ta cố tình truy cập vào một biến trong hàm hoặc phương thức, nhưng không thiết lập giá trị cho biến.

5.4.2. Sử dụng try ... except trong việc xử lý ngoại lệ

5.4.2.1. Cú pháp

try

khôi lệnh có khả năng xảy ra lỗi

except loại_lỗi_1 **as** tên_biến_báo_lỗi_1

in thông báo lỗi

except loại_lỗi_1 **as** tên_biến_báo_lỗi_1

in thông báo lỗi

...

else

khôi lệnh khi không có exception nào xảy ra

Lưu ý: có thể sử dụng lệnh *raise* để định nghĩa bổ sung cho *Exception*

5.4.2.2. Một số ví dụ

- Ví dụ 5.3: sử dụng *Exception* với *try ... except* để thực hiện yêu cầu chỉ cho nhập số nguyên dương. Kết quả được in ra màn hình tùy thuộc dữ liệu nhập vào của người dùng.

*mới học
thì xài
cách này*

```
def NhapSo():
    while True:
        try:
            n=eval(input('Nhập số nguyên >0: '))
        except Exception:
            print('Giá trị nhập không phải kiểu số')
        else:
            if type(n) is not int:
                print('Chỉ nhận số nguyên >0')
            elif n<=0:
                print('Là số nguyên nhưng phải >0')
            else:
                return n
print ('\nn=',NhapSo())
```

- *Ví dụ 5.4:* sử dụng `ZeroDivisionError` với `try ... except` để kiểm tra mâu số trong phép chia phải khác zero (0). Nếu mâu số (y) có giá trị khác 0, chương trình sẽ in r kết quả của phép chia x cho y, ngược lại sẽ xuất hiện lỗi `division by zero`

```
x, y = 5, 0
```

```
try:
```

```
    print(x, '/', y, '=', x/y)
```

```
except ZeroDivisionError as err:
```

```
    print('Error: ', err)
```

*lỗi được gán vào
tên biến là err*

- Ví dụ 5.5: tính tổng, hiệu, tích, thương của 2 số x và y. Sử dụng `ZeroDivisionError` với `try ... except ... finally` để kiểm tra mẫu số trong phép chia x cho y phải khác zero (0)

Mã lệnh	Kết quả
<pre>x, y = 5, 0 try: print(x, '/', y, '=', x/y) except ZeroDivisionError as err: print('Error: ', err) finally: <i>giống như else, sẽ là tiếp việc bên dưới</i> print(x, '+', y, '=', x + y) print(x, '-', y, '=', x - y) print(x, '*', y, '=', x * y)</pre>	<pre>Error: division by zero 5 + 0 = 5 5 - 0 = 5 5 * 0 = 0</pre>

- Ví dụ 5.6: sử dụng kết hợp *NameError* và *ZeroDivisionError*:

Mã lệnh	Kết quả
try: x = eval(input('input x: ')) y = eval(input('input y: ')) z = x/y except (NameError, ZeroDivisionError) as err: print('Error: ',err) else: print(x, '/', y, '=', z)	Ở ví dụ này: <i>bắt nhiều lỗi,</i> <i>bằng cách sử</i> <i>dụng dấu,</i> #Khi KHÔNG CÓ lỗi input x: >? 2 input y: >? 3 2 / 3 = 0.6666666666666666 #Khi CÓ lỗi input x: >? 4 input y: >? 0 Error: division by zero Hoặc: input x: >? 2

- Ví dụ 5.7: sử dụng kết hợp raise với ZeroDivisionError và NameError:

Mã lệnh	Kết quả
<pre> try: x = eval(input('input x: ')) y = eval(input('input y: ')) if (y==0): raise ZeroDivisionError ('y phải khác 0 (zero)') z = x/y except (NameError) as err: print('Error: ',err) except (ZeroDivisionError) as err: print('Error: ', err) else: print(x,'/',y,'=',z) </pre> <p style="color: blue; margin-left: 20px;"><Lưu ý> đây là cách thường được sử dụng</p> <p style="color: blue; margin-left: 20px;">các lỗi thường được bắt rời ra để xử lý</p>	<pre> #Khi KHÔNG có lỗi input x: >? 2 input y: >? 3 2 / 3 = 0.6666666666666666 #Khi CÓ lỗi input x: >? 4 input y: >? 0 Error: y phải khác 0 (zero) Hoặc: input x: >? 2 input y: >? a Error: name 'a' is not defined </pre>

mới học thì sử dụng cách này

```
def Nhapho():
    while True:
        try:
            n=eval(input('Nhập số nguyên >0: '))
        except Exception:
            print('Giá trị nhập không phải kiểu số')
        else:
            if type(n) is not int:
                print('Chỉ nhận số nguyên >0')
            elif n<=0:
                print('Là số nguyên nhưng phải >0')
            else:
                return n
print ('\nn=',Nhapho())
```

6/. Thực hiện mỗi yêu cầu sau đây thành một hàm chức năng. Viết chương trình gọi thực hiện các hàm này:

- a. Cho nhập 1 số nguyên dương n. Cần kiểm tra nếu n nhập vào không phải là kiểu số hoặc giá trị của $n \leq 0$, chương trình sẽ yêu cầu nhập lại cho đến khi nhập đúng. Hàm không có tham số đầu vào, sau khi nhập thành công, hàm trả về số nguyên n

Gợi ý: sử dụng `try ... except` để kiểm tra kiểu dữ liệu của n.

```
1 import random, math
2 def NhapSo():
3     while True:
4         try:
5             n = eval(input('Nhập số nguyên >0:'))
6         except Exception:
7             print('Giá trị không phải kiểu số')
8         else:
9             if type(n) is not int:
10                 print('Chỉ nhận số nguyên >0')
11             elif n <= 0:
12                 print('Là số nguyên nhưng phải >0')
13             else:
14                 return n
15 #-----
```

chương trình chính

```
#----- chương trình chính -----  
n=NhapSo()  
lst=Taolst(n)  
print("List ban dau:", lst)  
CauC(lst)  
CauD(lst)  
CauE(lst)
```

```
def Taolst(n):
    '''lst = []
    for i in range(n):
        lst.append(random.randint(0, 1001))
    return random.sample(range_(1,10000), n)
```

#-----

```
def CauC(lst):
    print("Cau C: cac cho chan:", end=' ')
    for i in lst:
        if i%2 == 0:
            print(i, end=' ')
        if i==99:
            print("99")
            break
#-----
```

```
def LaSoMayMan(k):
    while k>0:
        so=k%10
        if so !=6 and so !=8:
            return False
        k=k//10
    return True
```

```
#-----
```

```
def LaSoMayMan(k):  
    if k<=0: ←  
        return False  
    while k>0:  
        so=k%10  
        if so !=6 and so !=8:  
            return False  
        k=k//10  
    return True
```

*trong trường hợp
không muốn lá số
may mắn có chứa
số 0*

```
def CauD(lst):
    dem=0
    for i in lst:
        if LaSoMayMan(i):
            if dem==0:
                print("\nCác số may mắn có trong list là: ", end=" ")
                print("%5d"%i, end=" ")
                dem+=1
    if dem==0:
        print("\nTrong list không có số may mắn")
#-----
```

**biến đếm ở đây cho việc
nếu ko có bất kỳ số may
mắn nào trong list**

```
def ktraSNT(n):  
    if (n<=1):  
        return False  
    for i in range_(2,int(math.sqrt(n))+1):  
        if (n%i==0):  
            return False  
    return True
```

```
#-----
```

```
def CauE(lst):
    dem=0 ← sử dụng cho trường hợp
    for i in lst: không có bất kỳ số nguyên
        if ktraSNT(i): tố nào trong list
            if dem==0:
                print("\nCác số nguyên tố có trong list là: ", end=" ")
            print("%5d"%i, end=" ")
            dem+=1
    if dem==0:
        print("\nTrong list không có số nguyên tố")
```

3.5.2. Module random

- *Công dụng:* Giúp tạo ra số ngẫu nhiên
- *Sử dụng:* import đầu chương trình `from random import *`

3.5.2.1. Hàm dùng cho số nguyên (integers)

Gồm 2 hàm: `randrange ([start,] stop [, step])`
 `randint (start, end)`

- Trong đó:
 - *start*: số nhỏ nhất có thể phát sinh ngẫu nhiên. Mặc định là 0
 - *stop*: với *stop-1* là số lớn nhất có thể phát sinh ngẫu nhiên.
 - *end* \approx *stop+1*. Do đó:
 - Hàm `randint (start, end)` trả về số ngẫu nhiên *n*, với `start<=n<=end`.
 - Việc sử dụng 2 hàm sau là tương đương nhau:
`random.randint (start, end) \approx randrange (start, stop+1)`
 - *step*: lựa chọn các số có dạng *start+ (step*k)* và nằm trong khoảng từ *start* đến *stop-1*). Mặc định *step=1*.

- Hàm randint(start, end) trả về số ngẫu nhiên n , với $\text{start} \leq n \leq \text{end}$.

- Việc sử dụng 2 hàm sau là tương đương nhau:

random.randint(start, end) \approx randrange(start, stop+1)

- step : lựa chọn các số có dạng $\text{start} + (\text{step} * k)$ và nằm trong khoảng từ start đến $\text{stop}-1$.
Mặc định $\text{step}=1$.

- Minh họa tham số step

Mã lệnh	Các số được chọn để phát sinh ngẫu nhiên
randrange(2, 10, 1)	2, 3, 4, 5, 6, 7, 8, 9
randrange(2, 10, 2)	2, 4, 6, 8
randrange(2, 10, 3)	2, 5, 8

- Ví dụ 3.49

Mã lệnh	Kết quả
from random import *\br/>i=0 mylist=[] while i<10: mylist.append(random.randint(1, 10)) print(mylist)	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

- Mình họa tham số step

Mã lệnh	Các số được chọn để phát sinh ngẫu nhiên
randrange(2, 10, 1)	2, 3, 4, 5, 6, 7, 8, 9
randrange(2, 10, 2)	2, 4, 6, 8
randrange(2, 10, 3)	2, 5, 8

- Ví dụ 3.49

Mã lệnh	Kết quả
from random import * i=0 mylist=[] while i<10: mylist.append(randrange(2,10,3)) i+=1 print(mylist)	[8, 8, 2, 2, 5, 2, 8, 5, 2, 8]
for i in range (1,10): print(randint(2,6), end=' ')	5 4 4 2 3 6 4 3 6

3.5.2.2. Hàm dùng cho số thực (float)

- *random()*
 - Trả về giá trị ngẫu nhiên kiểu float nằm trong khoảng [0.0, 1.0).
- *uniform(a, b)*
 - Trả về giá trị ngẫu nhiên n có kiểu là float nằm trong khoảng
 - $a \leq n \leq b$ khi $a \leq b$
 - và $b \leq n \leq a$ khi $a > b$
 - Giá trị điểm cuối *b* có thể có hoặc không có trong kết quả phát sinh số ngẫu nhiên tùy thuộc vào việc làm tròn trong phương trình $a + (b-a) * random ()$.

3.5.2.3. Hàm dùng cho tập hợp có thứ tự

- *choice(seq)*

- Trả về phần tử ngẫu nhiên trong một tập hợp *seq* không rỗng.
- Nếu tập hợp *seq* rỗng sẽ phát sinh lỗi *IndexError*.
- Ví dụ 3.50

Mã lệnh	Kết quả
<pre>from random import * i=0 mylist=[] while i<10: mylist.append(randint(0, 10)) i+=1 print(mylist) print(choice(mylist))</pre>	[5, 5, 2, 2, 6, 8, 0, 6, 6, 6] 6

- *sample(range(a, b), quantity)*

- Trả về *quantity* số ngẫu nhiên *n*, với $a \leq n < b$ và *n* không trùng nhau.
- Sẽ phát sinh lỗi *ValueError* khi $b - a > quantity$

- `sample(range(a, b), quantity)`

- Trả về *quantity* số ngẫu nhiên n , với $a \leq n < b$ và n không trùng nhau.
- Sẽ phát sinh lỗi *ValueError* khi $b - a > quantity$
- *Ví dụ 3.51: tạo ngẫu nhiên 2 list số nguyên. Tạo ra list thứ 3 chứa các số có trong cả 2 list trước đó*

Mã lệnh	Kết quả
from random import * listA = sample(range(1,15), 10) print ("listA= ",listA)	listA= [13, 3, 12, 8, 5, 7, 11, 6, 4, 1]
listB = sample(range(1,15), 5) print ("listB= ",listB)	listB= [4, 7, 13, 1, 14]
listResult = [i for i in listA if i in listB] print ("listResult= ",listResult)	listResult= [13, 7, 4, 1]
listD = sample(range(1,5), 10) print ("listA= ",listA)	ValueError: Sample larger than population or is negative

Buoi05 > temp.py

1: Project

Project ▾ Buoi05 D:\TTTH_KHTN\Python270
Bai1_Tr26.py
Bai2_Tr26.py
Bai3_Tr26.py
Bai49_Tr11.py
Bai61&62_Tr13.py
Bai63_Tr13.py
temp.py

External Libraries
Scratches and Consoles

Bai3_Tr26.py x temp.py x

```
2 lst=sample_(range(1,1000000), 10)
3 print("Mới tạo",lst)
4 for index in range (len(lst)):
5     lst[index]=lst[index]/12345
6 print("After",lst)
7
8
```

ví dụ tạo random
list số thực

Bai3_Tr26 x temp x

```
C "C:\Program Files\Python39\python.exe" "C:\Program Files\JetBrains\PyCharm Community Edition 2021.1.1\helpers\pycharm\ipython\ipython3_launcher.py" --no-banner
import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['D:\\TTTH_KHTN\\Python270\\Buoi05', 'D:/TTTH_KHTN/Python270/Buoi05'])

Python 3.9.5 (tags/v3.9.5:0a7dcbd, May 3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)]
Mới tạo [703253, 521440, 170879, 230988, 899940, 833438, 271059, 414667, 373916, 8547]
After [56.96662616443904, 42.238963142972864, 13.84196030781693, 18.71105710814095, 7]
```

In[3]: |

2: Favorites
Structure